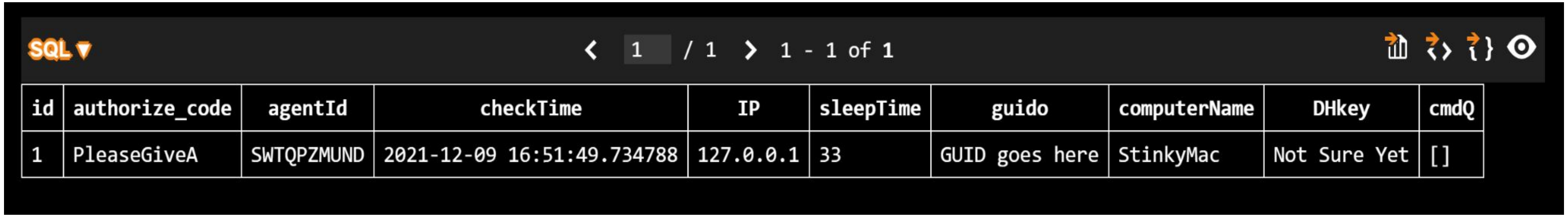
CS 501 Final Project

By: Akshey Nischal, Carlos Lopez, Jason Chow,
Jahyung Yun, and Alexandre Dennis

C2

- The C2 Server was built on Flask and hosted on a local machine and was given a DDNS
- But is ready and planned to be moved to a Raspberry pi
- Features
  - Uses MySQL as a database to store authorized agents, GUid, machine name, and task queue
  - Able to establish a communication path between c2 and implant and between c2 and client
- Features that are yet to be implemented
  - RSA encryption between c2 and implant
- Difficulties we ran into
  - Sending strings that could be parsed as JSON from the C++ implant to the Python C2
  - We got the private and public key generation working on the C2 but encoding a message with the public key on the implant kept giving errors



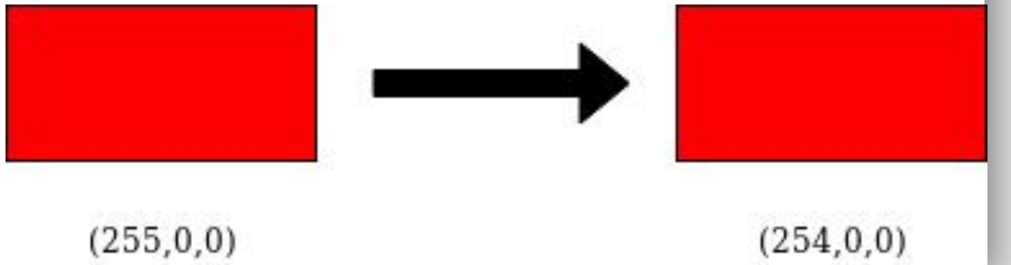
id	authorize_code	agentId	checkTime	IP	sleepTime	guido	computerName	DHkey	cmdQ
1	PleaseGiveA	SWTQPZMUND	2021-12-09 16:51:49.734788	127.0.0.1	33	GUID goes here	StinkyMac	Not Sure Yet	[]

Client

- Uses Python to communicate with C2
- Uses brand new pattern matching from Python 3.10 (released October this year)
- Can print out entire sqlite3 database holding computer and implant data
- Can send get or post requests to anywhere but defaults to C2 server
- Can send steganographic images to C2 server containing any sort of message
- Has default information and payloads to test C2 with

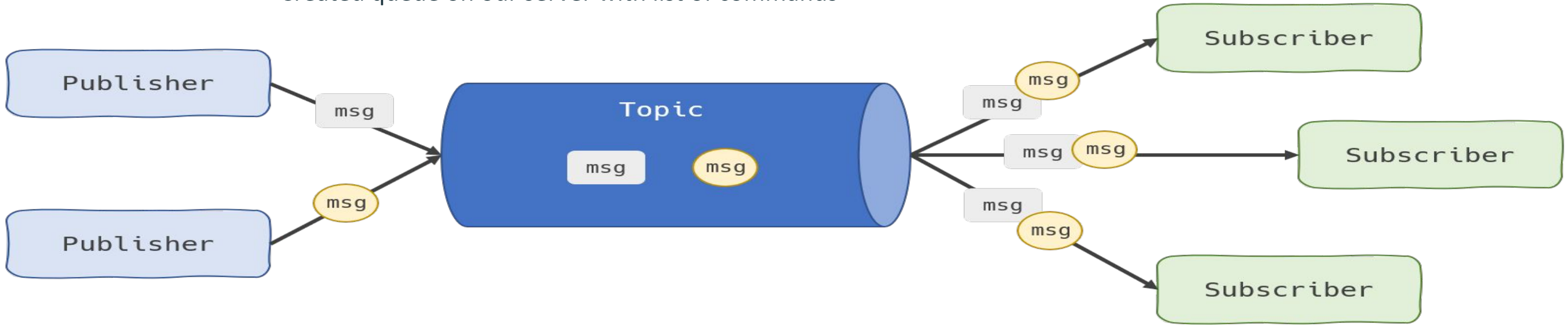
Special Feature (Steganography)

- The special feature we chose to implement is steganography
    - This is the action of hiding a message within another message or most commonly an image
- We were able to accomplish this using Least Significant Bit steganography
    - Accomplished by altering the values in the RGB matrix for each pixel (or for the required amount of pixels based on text length)
- As shown in the image below, a difference in value does necessarily produce a noticeable difference in image, therefore we are able to mask messages and greatly reduce the chance of detection
- As an added layer of security, we tried to implement an encryption scheme within the steganography function which secures our message in the event that the altered image is noticeably different



Pub Sub

- Form of asynchronous service-to-service communication used in serverless architectures.
- Any message published to a topic is immediately received by all of the subscribers to the topic.
- The issue
  - Initially tried it with Google Cloud
  - Permission / authorization issues
  - Endpoints and pushes
- The compromise (well…)
  - Created queue on our server with list of commands



Implants

- Features:
  - Checks for ch0nky.txt
  - Persistence- Adds itself to Registry key “HKEY_CURRENT_USERSoftware\Microsoft\Windows\CurrentVersion\Run”
  - Gets machine guid and name and sends it to c2
  - Request jobs/tasks from c2, executes it, and returns the output of those tasks back to c2
  - Task includes:
    - Stealer
    - Powershell command shells
- Features we are still working on:
  - DLL/ Reflective DLL injection
    - Plan for dropper: call final payload from c2, opening a process such as notepad or calc, injecting into the found process by checking PIDs
  - Public key encryption (specifically using RSA encryption scheme)
- Difficulties:
  - Our test.dll running successfully through bash but not terminal/powershell